dunetpc - Task #23726

Dataprep update

12/06/2019 01:06 PM - David Adams

Status:	Closed	**Start date:**	12/06/2019
Priority:	Normal	**Due date:**	
Assignee:	David Adams	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Dataprep run in production has not kept up with some of the changes I have been using in my studies and I would like update it.

These include:
1. Add the new bad and sticky codes shown at the last couple DRA meetings
2. Switch to the new tail removal
3. Ignore flagged sticky codes in the pedestal finder.

I put these all in one ticket because I would like to change one at a time and verify with the CI testing before proceeding to the next.

History

#1 - 12/06/2019 01:17 PM - David Adams

- Assignee set to David Adams

- Status changed from New to Work in progress

I start with the bad channels. The 17 bad channels show at the Nov 27 DRA meeting (https://indico.fnal.gov/event/22557/) are already in dunetpc. Before modifying the prolog channelstatus_pdsp in dune/Protodune/singlephase/fcl/channelstatus_pdsp.fcl, I copied it to pdsp_channel_status_2018 in pdsp_channel_status_2018.fcl in
case we want to go back to the old set.

Christoph reported a change between dunetpc revisions 9825e61b and 3f8748a that is presumably due to the new bad channels. We get 0.38% fewer hits in the data CI test presumably because we filter out bad channels prior to wirecell processing.

#2 - 12/27/2019 12:37 PM - David Adams

I have just pushed another bad channel update corresponding to my Dec 11 presentation. Here is a log snippet:

```
%MSG-i SimpleChannelStatusService:  Early 27-Dec-2019 12:06:07 CST JobSetup
Loaded from configuration:
  - 156 bad channels
  - 39 noisy channels
  - largest channel ID: 15359, largest present: 15359
```

As before, I would expect small changes in the wire containers in our test jobs. Christoph, please confirm that you see this. Thank you.

This is all the bad channels I know now.

#3 - 12/30/2019 05:48 AM - Christoph Alt

Yes, the datareco_protoDUNESP CI test does report a change in the hit collection size between dunetpc revision d69892ff and 5a96a6ef:

```
1252: < DecoderandReco | gaushit |  | std::vector<recob::Hit> | 46923

1256: > DecoderandReco | gaushit |  | std::vector<recob::Hit> | 46905
```

Full log: http://dbweb5.fnal.gov:8080/LarCI/app/ns:dune/storage/docs/2019/12/27/stdout%231qDDuMy.log

#4 - 12/31/2019 01:07 PM - David Adams

Thanks for the report. That small decrease sounds right.

#5 - 02/28/2020 07:39 AM - David Adams

I am back looking at this. I see two places where we define the standard dataprep tool sequences, tools.

`Utilities/services_dune.fcl`
`Utilities/services_refactored_pdune.fcl`

I think the definitions are the same there (same names and same sequences). Tingjun, can you explain the purpose of each of these files?

The duplication is not a good idea because one will have to remember to make changes in both places and users like myself are confused about which is actually include in protodune production reco. I propose to move these definitions to a dedicated file, maybe DataPrep/fcl/protototodune_dataprep_services.fcl that is then included in both of the above files.

Any objections?

#6 - 02/28/2020 01:25 PM - David Rivera

Hi David,

`Utilities/services_refactored_pdune.fcl`

is what we use in the refactored simulations for PDSP.

I agree with your suggested change of creating a dedicated file for dataprep to include in both service configuration files. Thanks for noticing this.

#7 - 03/02/2020 09:13 AM - David Adams

I am working on this now. The service configs are being moved to

`dunetpc/dune/DataPrep/fcl/protototodune_dataprep_services.fcl`

I am testing that the dataprep results for one data event are unchanged.

I am also moving the simulation configuration protodune_dataprep_tools_sim. Can someone point me to a simulation file I can use to test reco there?

Thanks.

da

#8 - 03/02/2020 09:35 AM - Tingjun Yang

Here is a MC file:
xroot://fndca1.fnal.gov:1094/pnfs/fnal.gov/usr/dune/tape_backed/dunepro/physics/detector-simulated/2019/mc/out1/PDSPProd2/22/60/37/10/PDSPProd2_protoDUNE_sp_detsim_p1GeV_35ms_sce_datadriven_22603710_127_373eda75-9c10-48f7-bf6c-83cabc4d64b9.root

The standard MC reco fcl file is:
protoDUNE_reco_35ms_sce_datadriven.fcl

#9 - 03/02/2020 09:39 AM - David Adams

As expected, the new reco looks identical to the old. Compare the new (reco_dataprep.1) and old (reco_dataprep.0) at https://internal.dunescience.org/people/dladams/protodune/test/test15.

I have pushed the mods to take both data and sim from the new config file.

Christoph, when the next CI tests run, could you let us know if any changes appear? Thank you.

#10 - 03/02/2020 09:44 AM - David Adams

Thanks Tingjun. I assume the CI will cover it but I will also do my own check of the sim data using your file.

#11 - 03/02/2020 10:04 AM - David Adams

I confirm the new sim (reco_dataprep_sim.1 at the same URL as above) is the same as the old (reco_dataprep_sim.0).

#12 - 03/02/2020 10:12 AM - David Adams

Changing gears for a minute, I report that PDSP channel 325 (APA 3 u) was added to the 2018 and 2019 noisy channel lists last week. Its pedestal is mostly stuck about 8 ADC counts from occasional forays to the presumably true pedestal position.

A typical waveform can be seen here:
https://internal.dunescience.org/people/dladams/protodune/data/wfRaw/run008564/event000001/apa3u/wfraw_run008564_evt000001_chan00320.png

#13 - 03/03/2020 02:59 AM - Christoph Alt

The CI test doesn't show changes in data product sizes.

#14 - 03/05/2020 10:24 AM - David Adams

Christoph, thanks for confirming the last change had no effect.

I have committed another change: the flagging of sticky codes (not the mitigation) is moved to the start of the dataprep tool sequence. This is expected to have no effect on reco and the plots at
https://internal.dunescience.org/people/dladams/protodune/test/test15
indicate this is the case.

To be very careful, I will wail for confirmation no change is seen in the CI tests before moving on to the next change.

#15 - 03/05/2020 11:36 AM - Tingjun Yang

Unfortunately I made another change to hit finder which changed to reconstruction result. Also larsoft version is changed so CI test does not work.

I have backed out my change. Once Christoph updated the dunetpc dependence on larsoft, we can check if David's change has any impact.

#16 - 03/06/2020 11:05 AM - Christoph Alt

The CI test does report a small change in the hit collection and downstream reco:
http://dbweb5.fnal.gov:8080/LarCI/app/ns:dune/storage/docs/2020/03/06/stdout%23grwqSOy.log

1259: < DecoderandReco | gaushit | | std::vector<recob::Hit> | 47091
1260: ---
1261: > DecoderandReco | gaushit | | std::vector<recob::Hit> | 47065

This is after moving to larsoft v08_45_00. I don't see changes in the hit finder from larsoft v08_44_00 to larsoft v08_45_00 or other changes in dunetpc that could explain the change in the hit collection, so this is likely due to your commit. You could back your commit out if you want to confirm this.

#17 - 03/10/2020 10:40 AM - Tingjun Yang

Hi David,

Are those changes expected?

#18 - 03/10/2020 11:59 AM - David Adams

No, I did not expect any changes. I will follow Christoph's suggestion and back the change out. --david

#19 - 03/10/2020 12:16 PM - David Adams

I take it back. The pedestal fitter is configured to ignore sticky codes (I forgot about that) and so small changes from the switch are to be expected. The fitter tries to identify and remove sticky codes before fitting but may not always succeed and cannot remove more than one. I believe all is OK now.

#20 - 03/10/2020 12:18 PM - Christoph Alt

I will update the reference files.

#21 - 09/04/2020 01:20 PM - David Adams

As discussed at 8/26 DRA, we would like to correct calibration to account for new measurement of injection capacitance and for estimated out-of-plane charge sharing. A new tool is added AdcRangeSampleScaler with configuration pdsp_gainCorrection is added to accomplish this. The PDSP data dataprep sequences have been updated to include this correction.

The change is pushed to dunetpc.

I have only tested to verify that the tool is run successfully and will do more testing tonight. We should expect a few more hits and slightly higher signals as the gain has been increased by about 5%. Please report here if any changes (good or bad) are seen.

#22 - 09/04/2020 05:39 PM - David Adams

I confirmed with signal strength plots that the scale change is as expected.

#23 - 09/05/2020 02:43 AM - Christoph Alt

The datareco_protoDUNESP CI test reports a small increase in reconstructed hits from 45762 to 46343. I will update the reference files.

#24 - 10/20/2020 07:40 AM - David Adams

Tingjun is ready to move to the new tail remover:

```
Ok, Thanks David. [In response to my proposal to move now to the new tool.]

Also according to the ProtoDUNE performance paper:
```

```
In each TPC channel, the amplifier and ADC are AC-coupled using a high-pass RC filter with a time constant of 
approximately τRC =1.1 ms (2200 ticks) for collection-plane channels and 3.3 ms (6600 ticks) for induction-pla
ne channels.

Could you update the tool to use 1.1 ms (2200 ticks)? Also could you apply the tool on all collection wires (b
oth TPC side and cryostat side).

Thanks,
Tingjun
```

My response to the above:

```
My long term plan was/is to carry out studies to see if there is are benefits to fine tuning the the RC consta
nt and to applying the correction to induction planes and cryostat-site collection. As is often the case, othe
r studies have preempted this. The change to TPC-side collection was made because the improvement was obvious 
by eye.

The direct effect of tail removal in the collection planes on an isolated signal is small: a 1-2% shift in are
a if the pedestal is evaluated in regions unaffected by the pedestal and somewhat less when those are included
. The effect is much bigger when a signal of interest falls in the tail of a much larger signal.

Switching to 1.1 ms will decrease the maximal isolated signal shift  by 0.015%.

I will make the changes you requested.
```

#25 - 10/20/2020 08:37 AM - David Adams

- File adcprp_tpp0z_run000001_evt000001.png added

- File adcprp_tpp0z_run000001_evt000001.png added

The above changes are in dunetpc. Here is the tail removal configuration:

```
  pdspTailPedRemovalZKe: {
    DecayTime: 2200
    ExcludeChannelRanges: []
    IncludeChannelRanges: ["apa1z", "apa2z", "apa3z", "apa4z", "apa5z", "apa6z", "apa1c", "apa2c", "apa3c", "a
pa4c", "apa5c", "apa6c"]
    LogLevel: 1
    MaxTick: 6000
    PedDegree: 0
    PedFreqs: []
    PedTick0: 3000
    SignalFlag: 3
    SignalIterationLimit: 10
    SignalTool: "adcTailSignalFinderKe" 
    tool_type: "ExpTailPedRemover" 
  }
```

This tool is now used by defaults for both data and simulation. I attach displays showing one plane without and with this tail removal.

#26 - 10/20/2020 08:39 AM - David Adams

Changes are committed to dunetpc.

Etienne or Tingjun, could you confirm that next round of CI tests doesn't show any problems?

Thank you.

#27 - 10/26/2020 04:59 PM - David Adams

I noticed the new tail remover was emitting some warning messages (grep "WARNING:"). These were for bad channels. I modified the messages to show the channel status and added config parameter (NoWarnStatuses) that can be used to suppress the messages for specified status flags. The default config suppresses them for bad channels so we no longer see the messages.

#28 - 10/26/2020 05:04 PM - David Adams

The dataprep service now produces a report at the end of job showing the time used in each of the dataprep tools. E.g. in the current version of my test job:

```
ToolBasedRawDigitPrepService:dtor: Event count: 1
ToolBasedRawDigitPrepService:dtor:  Call count: 6
```

```
ToolBasedRawDigitPrepService:dtor: Time report for 8 tools.
ToolBasedRawDigitPrepService:dtor:                   digitReader:   0.08 sec/event
ToolBasedRawDigitPrepService:dtor:          pdsp_sticky_codes_ped:   0.04 sec/event
ToolBasedRawDigitPrepService:dtor:              pd_adcPedestalFit:   6.02 sec/event
ToolBasedRawDigitPrepService:dtor:           adcSampleCalibration:   0.37 sec/event
ToolBasedRawDigitPrepService:dtor:            pdsp_gainCorrection:   0.04 sec/event
ToolBasedRawDigitPrepService:dtor:               pdsp_adcMitigate:   0.38 sec/event
ToolBasedRawDigitPrepService:dtor:            pdsp_timingMitigate:   0.01 sec/event
ToolBasedRawDigitPrepService:dtor:          pdspTailPedRemovalZKe:   3.34 sec/event
```

#29 - 10/26/2020 05:15 PM - David Adams

All changes are pushed to the head of dunetpc. Please let me know right away if there are any problems. If not, I will close this ticket soon.

#30 - 10/27/2020 05:18 AM - Etienne Chardonnet

Hi David,

Nothing alarming on the CI tests (from 46238 to 46479 hits). I will update the ref file

#31 - 10/27/2020 07:12 AM - David Adams

- Status changed from Work in progress to Closed

Thank you.

#32 - 10/27/2020 12:28 PM - David Adams

I added the Fcl item CallgrindToolNames to ToolBasedRawDigitPrepService. By default, it is everywhere (in dunetpc), set to []. If tool names are added to this list, valgrind profiling (if run) is toggled on and off when each tool is run.

#33 - 10/27/2020 04:27 PM - David Adams

- Status changed from Closed to Work in progress

I have made a small optimization in the tail remover: symmetric elements of the matrix are copied instead of calculated. There should be no change in results and the tool should run 10% faster. Pls let me know if you see any changes. I just pushed the mod.

I did check that the unit test gives the same result.

#34 - 01/18/2021 01:00 PM - David Adams

- Status changed from Work in progress to Closed

Presumably this is fine.

Files

adcprp_tpp0z_run000001_evt000001.png	1020 KB	10/20/2020	David Adams
adcprp_tpp0z_run000001_evt000001.png	1.01 MB	10/20/2020	David Adams